Modélisation et contrôle d'un serveur

Luc Malrait, Sara Bouchenak, Nicolas Marchand

RenPar'19 / SympA'13 / CFSE'7
Toulouse, 9 au 11 septembre 2009

Modélisation et contrôle d'un serveur

Luc Malrait[1], Sara Bouchenak[2], Nicolas Marchand[3]

Résumé
Dans cet article nous étudions l'influence de la configuration d'un serveur sur les aspects antinomiques de performance et de disponibilité du service. Dans un premier temps nous présentons un modèle de serveur construit à partir d'approximations de type fluide. Ensuite nous développons deux lois de contrôle d'admission qui visent à optimiser la configuration du serveur afin de respecter deux objectifs de QoS. Le premier objectif consiste à maximiser la disponibilité de service tout en garantissant une contrainte sur sa performance. Le second consiste à maximiser la performance du service tout en garantissant une contrainte sur sa disponibilité. Le modèle et les lois de commandes sont évaluées expérimentalement sur le banc d'essai TPC-C.

1. Introduction

Il existe de nombreux services Internet : les serveurs web, les serveurs de messagerie [28], les services de transmission multimedia en continu [3] , les serveurs commerciaux [2] ou encore les serveurs de base de données [24]. Ces services reposent le plus souvent sur l'architecture client-serveur, dans laquelle plusieurs clients accèdent concurremment à un service en ligne fourni par un serveur (comme la lecture d'une page web, l'envoi d'emails ou le paiement d'un panier d'achats). De tels systèmes de serveur sont soumis à des variations de charge, comme on peut le voir dans différentes études [5, 8, 4]. Un serveur de messagerie est par exemple soumis à une plus forte charge le matin que le reste de la journée, puisqu'il est commun de consulter ses emails en arrivant au travail. Dans le cas extrême, une forte charge peut provoquer un écroulement du serveur et entraîner une indisponibilité du service. Les coûts engendrés par ce phénomène sont estimés à plus de 2 millions de dollars par heure pour les entreprises de télécommunication et les entreprises financières [13, 23].
Une technique classique pour empêcher les serveurs de s'écrouler quand la charge devient trop importante est le contrôle d'admission [12]. Cela consiste à limiter le nombre de clients en concurrence sur le serveur, c'est à dire à fixer le niveau maximal de multiprogrammation du serveur (MPL). Il est clair que le choix du MPL d'un serveur a des conséquences directes sur sa performance, la disponibilité de son service et sa qualité de service (QoS). Les approches existantes de contrôle d'admission reposent sur un réglage ad-hoc, sur des heuristiques sans garantie d'optimalité [6, 22, 21], utilisent la théorie de la commande linéaire, qui ne peut pas capturer le comportement intrinsèquement non linéaire des systèmes de serveur, [26, 7, 11], ou bien appliquent la théorie des files d'attente, grâce à laquelle le système peut être modélisé de manière précise, mais en dépit d'une grande difficulté à le paramétrer, ce qui le rend peu utilisable [29, 31, 25].
Nous croyons en la nécessité de modéliser les systèmes de serveur afin de garantir différents critères de QoS. Cependant, il paraît clair que l'effort de modélisation doit être orienté de telle sorte que les modèles capturent la *dynamique* et le comportement *non linéaire* des serveurs tout en étant simple à déployer sur un système réel.

Contributions scientifiques

Les principales contributions de cet article sont les suivantes :
– La construction d'un modèle dynamique non linéaire de serveur capable de reproduire le comportement moyen d'un système réel aussi bien à faible qu'à forte charge.
– La construction et l'implémentation d'un contrôle d'admission pour un serveur. Deux lois de commandes en boucle fermée sont proposées : AM-$\mathscr{C}$ garantit une contrainte de performance en maximisant la

[1]INRIA – Gipsa Lab, Luc.Malrait@inria.fr
[2]Univ. of Grenoble I – INRIA, Sara.Bouchenak@inria.fr
[3]CNRS – Gipsa Lab, Nicolas.Marchand@inria.fr

disponibilité du service, PM-$\mathscr{C}$ garantit une contrainte de disponibilité du service en maximisant la performance.

Des expérimentations sur l'application TPC-C, un banc d'essai industriel répandu, associée au serveur de base de données PostgreSQL, sont présentées. Les résultats montrent l'amélioration significative des performances du sytème lorsqu'on lui applique un contrôle en boucle fermée.

Plan de l'article

La suite de l'article est organisée comme suit. La section 2 présente le contexte des travaux. Les sections 3 et 4 détaillent nos contributions en terme de modélisation et de contrôle en boucle fermée. La section 5 décrit les résultats de l'évaluation expérimentale, et la section 6 présente les principaux travaux apparentés. Enfin, la section 7 expose nos conclusions.

2. Contexte

2.1. Systèmes de serveurs

Nous considérons des sytèmes de serveur tels que les serveurs de base de données et les serveurs web qui suivent l'architecture client-serveur. Les clients et les serveurs sont hébergés par différentes machines connectées entre elles par un réseau de communication. Typiquement, un client se connecte à distance à un serveur, envoie sa requête, le serveur traite la requête et construit la réponse. Cette dernière est envoyée au client qui ferme sa connection. Plusieurs clients peuvent accéder concurremment au même serveur.

La charge du serveur

Elle est caractérisée d'une part par le nombre de clients qui essaient d'accéder concurremment au serveur (i.e. la quantité de charge), et d'autre part par la nature des requêtes soumises par les clients (i.e. le mix de charge). On peut par exemple trouver des requêtes en lecture seule ou en lecture-écriture, plus ou moins consommatrices de ressources processeur ou disque. La quantité de charge est notée N et le mix de charge M. Cette charge peut bien sûr varier au cours du temps, selon le comportement des clients.

Le contrôle d'admission

C'est une technique classique pour empêcher un serveur de s'écrouler quand le nombre de clients concurrents devient trop élevé [12]. Cela consiste à fixer une limite sur le nombre de clients qui peuvent potentiellement utiliser le serveur en même temps. Cette limite est un paramètre de configuration réglable du serveur que l'on appelle niveau de multiprogrammation (MPL). Au delà de cette limite les requêtes entrantes des clients sont rejetées. Il s'en suit que parmi les N clients qui essaient d'interagir au même moment avec le serveur, seulement N_e accèdent réellement à son service, et la contrainte $N_e \leq MPL$ est toujours respectée. Le MPL d'un serveur a donc une influence directe sur sa performance, sa disponibilité et sa QoS.

2.2. Performance et disponibilité de service

Plusieurs critères peuvent être considérés pour caractériser la performance et la disponibilité d'un service [21]. Dans la suite, nous considérons en particulier deux grandeurs importantes du point de vue utilisateur, qui sont la *latence* et le *taux de rejet*.

Performance du service – Latence

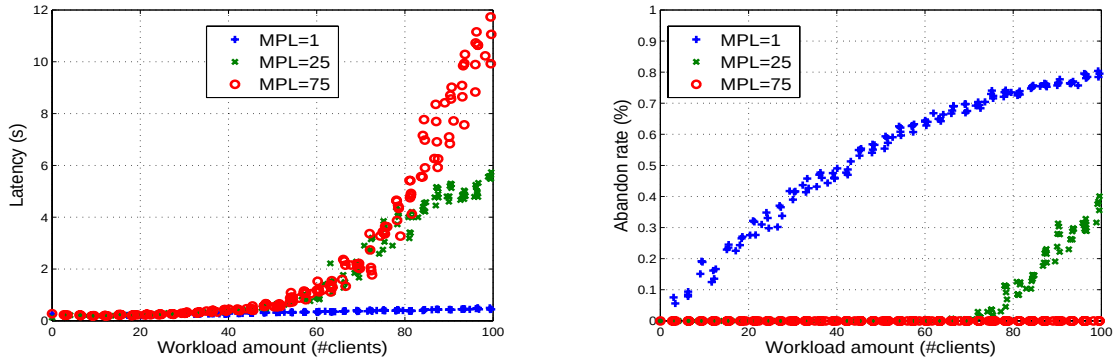
La latence d'une requête client est définie comme le temps nécessaire au serveur pour la traiter. La latence moyenne est notée L. Une faible latence est préférable, cela reflète la rapidité du système. La figure 1(a) décrit l'influence du MPL sur la latence quand la quantité de charge varie[1]. Ici, trois valeurs de MPL sont considérées : une faible (1), une moyenne (25) et une grande (75). Un MPL faible maintient le serveur en sous charge, ce qui implique une faible latence, tandis qu'un MPL élevé n'empêche pas une surcharge du serveur, qui peut induire une latence élevée.

Disponibilité de service – Taux de rejet

Le taux de rejet des requêtes clients est défini comme le rapport entre les requêtes rejetées à cause du contrôle d'admission et le nombre total de requêtes soumises au serveur. Il est noté α. Un faible taux de

[1] Des détails sur la plateforme expérimentale sont donnés section 5.1.

rejet est préférable, cela reflète de la bonne disponibilité du service. La figure 1(b) décrit l'influence du MPL sur le taux de rejet. Un MPL faible implique une concurrence forte pour accéder au serveur, ce qui se traduit par un taux de rejet plus élevé que pour un MPL élevé.



(a) Influence du MPL sur la performance

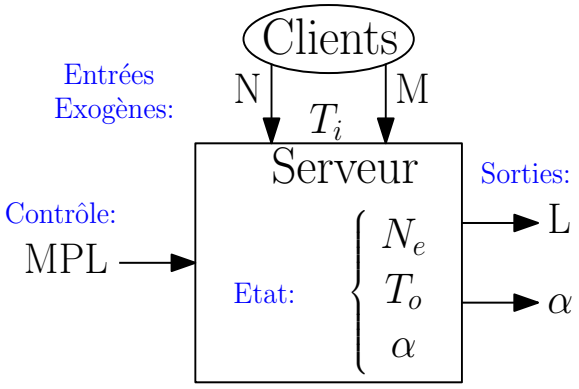
(b) Influence du MPL sur la disponibilité

FIG. 1 – Influence du MPL sur la performance et la disponibilité

La performance et la disponibilité d'un service font partie du SLA (Service Level Agreement). Le SLA spécifie les objectifs de niveau de service (SLOs) tels qu'une latence maximale L_{max} et un taux de rejet maximal α_{max} qui doivent être garantis par le serveur.

3. Modèle fluide de serveur

Nous proposons un modèle fluide qui exprime la dynamique d'un serveur et qui capture les caractéristiques qui reflètent son état en terme de performance et de disponibilité. On peut considérer qu'une approximation de type fluide consiste à voir les variables d'état du système, qui sont pour la plupart entières, comme des variables réelles dans $\mathbb{R}$. Cela permet de décrire des variations infinitésimales des grandeurs caractéristiques du système en fonction du temps. Ces variations peuvent être vues comme des flux de liquide (un flux de requêtes dans le cas présent), et les files d'attente sur le serveur comme un réservoir [1]. Le modèle est alors présenté comme un système d'équations différentielles, comme pour la plupart des systèmes physiques, qui décrit l'évolution temporelle de variables d'état.



Dans le cas présent, nous identifions trois grandeurs d'état qui décrivent et qui ont une influence sur la performance et la disponibilité du serveur : le nombre courant de requêtes en concurrence sur le serveur N_e, le débit du serveur T_o et le taux de rejet α. Les entrées du modèle proposé sont : la quantité de charge à laquelle est soumis le serveur N et le mix de charge M qui sont exogènes, le MPL du serveur qui est un paramètre réglable pour le contrôle d'admission. En plus des variables d'état et d'entrée, le modèle possède des variables de sortie telles que la latence moyenne L. Dans la suite décrivons le modèle proposé à l'aide d'équations reliant les variables d'état entre elles.

La présence d'un contrôle d'admission implique que parmis les N clients en concurrence qui essaient de se connecter au serveur, seulement N_e y parviennent, avec $0 \leq N_e \leq N$ et $0 \leq N_e \leq MPL$. Soit $cr(t, t+dt)$ le nombre de connexions crées sur le serveur entre t and $t+dt$, et $cl(t, t+dt)$ le nombre de connexions fermées entre t et $t+dt$. Un bilan sur N_e entre t et $t+dt$ donne

$$N_e(t+dt) = N_e(t) + cr(t, t+dt) - cl(t, t+dt) \tag{1}$$

Soit T_i le débit d'entrée sur le serveur, considéré comme le nombre de demandes de connexion par seconde. En considérant T_i et α constants sur l'intervalle $[t, t+dt]$, il vient que le nombre de connexions crées entre t et $t+dt$ s'écrit

$$cr(t, t+dt) = (1 - \alpha(t)) \cdot T_i(t) \cdot dt \tag{2}$$

Où α est le taux de rejet du serveur. De la même manière, définissons T_o comme le débit de sortie du serveur, considéré comme le nombre de requêtes servies par seconde par le serveur. En considérant T_o constant sur $[t, t+dt]$, le nombre de connexions fermées entre t et $t+dt$ s'écrit alors

$$cl(t, t+dt) = T_o(t) \cdot dt \tag{3}$$

Nous déduisons de (1), (2) et (3) une expression de $\dot{N}_e$, la dérivée temporelle de N_e

$$\dot{N}_e(t) = (1 - \alpha(t)) \cdot T_i(t) - T_o(t) \tag{4}$$

De plus, nous faisons l'hypothèse que le système atteint son régime permanent en une courte durée Δ ; ce qui se reflète en particulier dans les variables T_o et α. Pendant cette durée, la charge est relativement stable, ce qui est en accord avec des études comme [4]. Par conséquent, les dynamiques de T_o et α peuvent être approchées par des systèmes du premier ordre

$$\dot{T}_o(t) = -\frac{1}{\Delta}\left(T_o(t) - \bar{T}_o\right)$$
$$\dot{\alpha}(t) = -\frac{1}{\Delta}\left(\alpha(t) - \bar{\alpha}\right)$$

où $\bar{T_o}$ et $\bar{\alpha}$ sont les valeurs en régime permanent respectivement du débit de sortie et du taux de rejet. L'étape suivante consiste à trouver une expression de $\bar{T_o}$ et $\bar{\alpha}$. Un bilan sur le nombre de requêtes servies N_o donne

$$N_o(t+dt) = No(t) + sr(t, t+dt)$$

où $sr(t, t+dt)$ est le nombre de requêtes servies entre t et $t+dt$. Puisqu'il y a N_e requêtes concurrentes sur le serveur et que la latence moyenne est L, le nombre de requêtes servies pendant dt sera $sr(t, t+dt) = \frac{dt}{L} N_e$. D'où $\dot{N_o} = \frac{N_e}{L}$, i.e. $\bar{T_o} = \frac{N_e}{L}$ qui est une expression de la loi de Little [18].

Par définition, $\bar{\alpha}$ vaut zéro si N_e est inférieur au MPL, et $\bar{\alpha}$ vaut $1 - \frac{T_o}{T_i}$ si $N_e = MPL$ (voir Figure 2(a), naive model). Cependant, la nature aléatoire de l'instant d'arrivée des requêtes peut conduire à une situation dans laquelle le N_e moyen mesuré est inférieur au MPL mais ponctuellement, le nombre de clients qui essaient d'accéder au serveur est supérieur au MPL, ce qui implique des rejets. Ceci est illustré dans la figure 2(a) où le taux de rejet mesuré est comparé avec son estimation naïve décrite plus haut, et où l'on voit une différence entre les deux. Afin de tenir compte de ce comportement, nous choisissons d'écrire $\bar{\alpha} = \frac{N_e}{MPL} \cdot \left(1 - \frac{T_o}{T_i}\right)$. Cela traduit que la probabilité de rejeter un client est plus grande lorsque N_e moyen est proche du MPL. La figure 2(a) montre que cette méthode donne une estimation plus précise du taux de rejet. Finalement on obtient

$$\dot{T}_o(t) = -\frac{1}{\Delta}\left(T_o(t) - \frac{N_e(t)}{L(t)}\right) \tag{5}$$

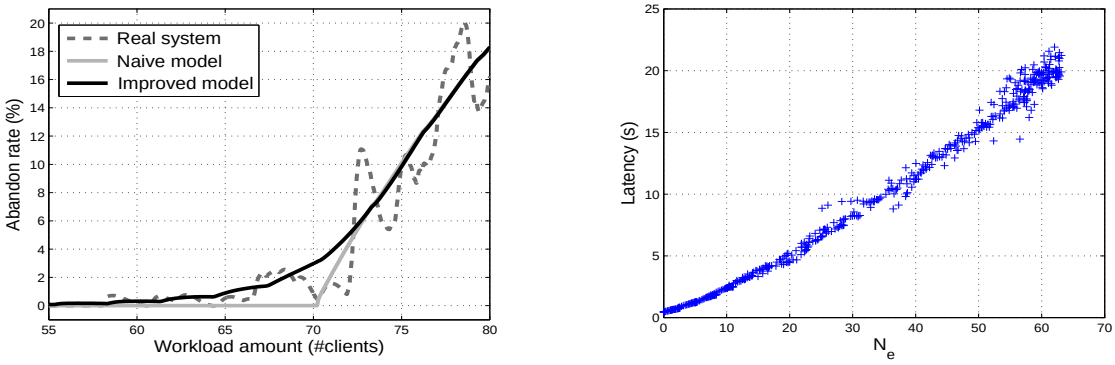
$$\dot{\alpha}(t) = -\frac{1}{\Delta}\left(\alpha(t) - \frac{N_e(t)}{MPL(t)} \cdot \left(1 - \frac{T_o(t)}{T_i(t)}\right)\right) \tag{6}$$

Maintenant que les variables d'états du modèle sont définies, il ne reste plus qu'à exprimer la variable de sortie du modèle L. La latence dépend évidemment de la charge du serveur, en particulier du mix de charge M, et du nombre de clients concurrents sur le serveur N_e. La figure 2(b) décrit l'évolution de la latence L

en fonction de N_e, pour un mix de charge donné. On peut voir qu'un polynôme du second degré en N_e est une bonne approximation de la latence L. On écrit alors

$$L(N_e, M, t) = a(M,t)N_e^2 + b(M,t)N_e + c(M,t) \tag{7}$$

Le paramètre c est positif puisqu'il représente la latence à charge nulle. a et b sont aussi positifs puisqu'ils modélisent le coût de traitement en temps.



(a) Précision du taux de rejet modélisé

(b) Latence en fonction de N_e

FIG. 2 – Modélisation du taux de rejet et de la latence

En résumé, le modèle proposé est décrit par les équations (4) à (7) qui traduisent la dynamique de l'état du serveur en terme de performance et de disponibilité. La section 4 décrit des lois de commande construites sur ce modèle dans le but de garantir des objectifs de performance et de disponibilité pour le service.

4. Contrôle de serveur

Dans la suite nous étudions le compromis entre performance et disponibilité des serveurs. A partir du modèle proposé, nous construisons un contrôle d'admission optimal en rapport avec ce compromis. Nous proposons en particulier deux lois de commande, nommées AM-$\mathscr{C}$ and PM-$\mathscr{C}$. AM-$\mathscr{C}$ garantit une contrainte de performance en maximisant la disponibilité du service et PM-$\mathscr{C}$ garantit une contrainte de disponibilité du service en maximisant la performance. Dans le cas présent, la disponibilité de service est mesurée par le taux d'acceptation des requêtes (i.e. 1-α), et la performance du service est mesurée par la latence moyenne (i.e. L).

4.1. AM-$\mathscr{C}$ commande en disponibilité maximale

AM-$\mathscr{C}$ a pour but de garantir le compromis entre performance et disponibilité de service dont les propriétés sont :
(P1) la latence moyenne n'excède pas une latence maximale L_{max}
(P2) le taux de rejet α est rendu le plus petit possible.
Pour cela, une loi de commande qui ajuste automatiquement le MPL du serveur est proposée. L'idée de base est d'asservir la latence L à L_{max} en admettant le nombre de client nécessaire. Par construction cela maximise le nombre de clients admis N_e, et minimise le taux de rejet α.
Une première approche pourrait consister à résoudre l'équation (7) de telle sorte que $L = L_{max}$. Elle est précise mais requiert la connaissance en temps réel des paramètres a, b et c dans l'équation 7, puisque la charge peut varier dans le temps. Une identification en ligne est possible mais nous privilégions une approche qui ne nécessite pas de connaître ces paramètres. Elle consiste en une linéarisation entrée sortie dans laquelle la sortie considérée est la latence L [15]. Pour résumer il s'agit de déterminer le MPL de telle sorte que la latence suive

$$\dot{L} = -\gamma_L (L - L_{max}) \tag{8}$$

Pour $\gamma_L > 0$, la convergence de L vers L_{max} est assurée. De l'équation (7) nous déduisons $\dot{L} = (2aN_e + b)\dot{N}_e$. Comme T_o et α atteignent leur régime permanent en une courte durée, $T_o(t) = \bar{T}_o$ et $\alpha(t) = \bar{\alpha}$. Par consé-

quent, avec l'équation (4) on obtient

$$\dot{L} = (2aN_e + b)\left(1 - \frac{N_e}{MPL}\right)\left(T_i - \bar{T}_o\right) \tag{9}$$

Des équations (8) et (9) on déduit le MPL

$$MPL = \frac{N_e}{1 + \frac{\gamma_L}{(2aN_e+b)(T_i-\bar{T}_o)}(L - L_{max})}$$

Afin de nous affranchir des paramètres a et b, nous choisissons d'utiliser $\gamma'_L = \frac{\gamma_L}{(2aN_e+b)(T_i-\bar{T}_o)}$, ce qui donne

$$MPL = \frac{N_e}{1 + \gamma'_L(L - L_{max})} \tag{10}$$

où $\gamma'_L > 0$ est un paramètre réglable. Il s'en suit qu'avec l'équation (9) et la commande (10), l'évolution dynamique de L est donnée par :

$$\dot{L} = -\left(\gamma'_L\left(2aN_e + b\right)\left(T_i - \bar{T}_o\right)\right)(L - L_{max})$$

Là encore, la convergence de L vers L_{max} est assurée.
En résumé, il est intéressant de noter que la loi de commande donnée en (10) traduira une des situations suivantes. Si à un instant donné la latence L est supérieure à L_{max}, la propriété (P1) n'est pas satisfaite, et alors le MPL prendra une valeur inférieure au nombre N_e de clients admis (puisque $(1 + \gamma'_L(L - L_{max})) > 1$), ce qui va dans le sens du respect de (P1). De la même manière, si L est inférieure à L_{max}, (P1) est respectée alors que (P2) peut ne pas l'être. Le MPL prendra alors une valeur supérieure à N_e (puisque $(1 + \gamma'_L(L - L_{max})) < 1$), ce qui va dans le sens du respect de (P2). Enfin si L vaut L_{max}, les propriétés (P1) et (P2) sont respectées.

4.2. PM-$\mathscr{C}$ commande en performance maximale

De même que précédemment, PM-$\mathscr{C}$ a pour but de garantir le compromis suivant :
(P3) le taux de rejet ne dépasse pas un taux de rejet maximum α_{max},
(P4) la latence moyenne est la plus petite possible
Dans ce cadre, en supposant que (P3) est respectée, (P4) sera satisfaite ssi le MPL converge vers la plus petite valeur qui garantisse $\alpha \leq \alpha_{max}$. Là encore une approche par linéarisation entrée-sortie est employée, en prenant α comme sortie.

$$\dot{\alpha} = -\gamma_\alpha(\alpha - \alpha_{max}) \tag{11}$$

avec $\gamma_\alpha > 0$. Comme il a été dit précédemment, puisque la charge reste stable pendant une courte durée, on a $\dot{N}_e = 0$. Donc des équations (4) et (6) on déduit

$$\alpha = 1 - \frac{T_o}{T_i}$$

$$\dot{\alpha}(t) = -\frac{1}{\Delta}\alpha(t)(1 - \frac{N_e(t)}{MPL(t)}) \tag{12}$$

Avec (11), (12) et le MPL suivant, on assure la convergence de α vers α_{max}

$$MPL = \frac{\alpha N_e}{\alpha + \gamma'_\alpha(\alpha - \alpha_{max})} \tag{13}$$

où $\gamma'_\alpha = \gamma_\alpha\Delta$.
En résumé les lois de contrôle d'admission proposées ne nécessitent qu'un seul paramètre externe qui est γ. Ce paramètre influe à la fois sur le temps de convergence de la commande (i.e. le nombre d'itérations nécessaire pour converger vers le MPL optimal) et sur la stabilité du système en boucle fermée. En effet un γ faible engendrera un temps de convergence long, tandis qu'un γ élevé pourra induire des oscillations dans le système bouclé. La partie délicate réside dans le choix de ce paramètre, qui découle en partie du temps nécessaire pour que le critère de QoS considéré (comme le taux de rejet dans le cas de PM-$\mathscr{C}$) atteigne son régime permanent.

5. Evaluation

Cette section décrit d'abord l'environnement applicatif de nos expérimentations avant de présenter les résultats de l'évaluation du modèle proposé et des lois de commande.

5.1. Contexte expérimental

Le banc d'essai.

L'évaluation du modèle proposé et des commandes en boucle fermée a été menée en utilisant le banc d'essai TPC-C [30]. TPC-C est un banc d'essai industriel standard, issu du "Transaction Processing Council", qui reproduit une application réaliste dans laquelle des "warehouses" hébergées par un serveur de base de données font l'objet de transactions soumises par des clients émulés. L'émulateur TPC-C permet de définir le nombre de clients concurrents à lancer (i.e. la quantité de charge N). Il permet aussi de définir le "think time", qui est la durée moyenne d'inter-arrivée entre deux requêtes consécutives. Nous avons étendu l'émulateur client afin de pouvoir faire varier au cours du temps la quantité de charge N d'une part, et le mix de charge M d'autre part. Pour cette dernière extension, nous avons considéré deux mix de charge, un constitué de requêtes en lecture seule et l'autre constitué de requêtes en lecture-écriture.

Environnements logiciel et matériel.

Nos expériences ont été menées sur un couple d'ordinateurs connectés par Ethernet à 100 Mb/s, un est dédié au serveur de base de données et l'autre à l'émulateur client. Le serveur est PostgreSQL 8.2.6 [24]. Le modèle proposé et les lois de commande ont été déployées comme suit. Des mesures en ligne sur le système sont menées afin de connaître à chaque instant l'état du modèle. Des techniques de filtrage de type Kalman sont appliquées pour obtenir des données exploitables [14]. Nous avons suivi une approche de type proxy pour implémenter les commandes AM-$\mathscr{C}$ et PM-$\mathscr{C}$. Le contrôle d'admission est fait sur le proxy situé devant le serveur. Dans les expériences suivantes, AM-$\mathscr{C}$ et PM-$\mathscr{C}$ont été appliquées avec $\gamma'_L = 0.1$ et $\gamma'_\alpha = 0.3$. Ces valeurs ont été choisies arbitrairement en respectant les contraintes de stabilité, à savoir $\gamma'_\alpha > 0$ et $\frac{1}{L_{max}} > \gamma'_L > 0$, L_{max} étant fixé plus tard à 8s. La machine serveur a un processeur de 3GHz avec 2GB de RAM, l'ordinateur clients a un processeur de 2GHz avec 512MB de RAM.

5.2. Validation du modèle

Nous effectuons des mesures pour valider la précision du modèle et sa capacité à exprimer les dynamiques du système. En particulier nous évaluons la capacité du modèle à faire évoluer ses grandeurs d'état, quand les entrées telles que le MPL ou la quantité de charge N varient, de manière cohérente avec l'évolution système réel. Nous nous intéresserons donc aux variables d'état qui sont : N_e pour le nombre de clients en concurrence sur le serveur, T_o pour le débit de sortie du serveur, et α pour le taux de rejet. Pour les mêmes variations des grandeurs d'entrée, nous comparons l'évolution des états du modèle et du système réel.

La figure 3 illustre le cas d'un système en boucle ouverte quand à la fois la quantité de charge N et le MPL du serveur varient au cours du temps (voir figure 3(a)). Les figures 3(b), 3(c) et 3(d) présentent l'évolution respective de N_e, T_o et α au cours du temps, pour le système réel (+) et le modèle (trait plein). On constate que le modèle traduit bien le comportement du système réel.

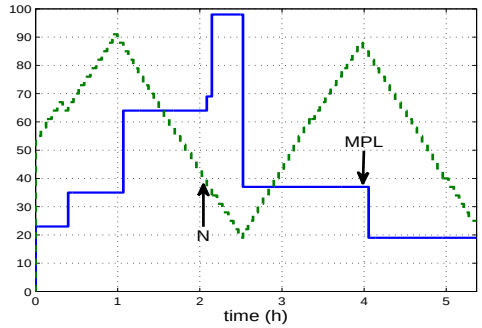
5.3. Evaluation du contrôle

Cette section présente les résultats de l'évaluation expérimentale des lois de commandes en boucle fermée présentées en 4. Les résultats obtenus avec la commande AM-$\mathscr{C}$sont présentés en 5.3.1, et ceux obtenus avec PM-$\mathscr{C}$en 5.3.2.

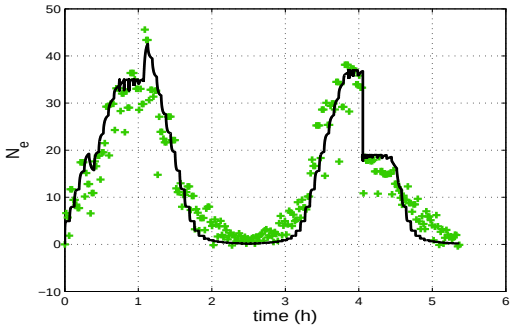
5.3.1. AM-$\mathscr{C}$ commande en disponibilité maximale

Ici, nous considérons une contrainte de performance qui limite la latence moyenne des requêtes à 8s. Le rôle de AM-$\mathscr{C}$est alors de garantir cette contrainte tout en maximisant la disponibilité du service, en ajustant en ligne le MPL. Nous utilisons deux scenarii pour évaluer cette commande. Chacun illustre une variation d'une des deux entrées éxogènes du système : le mix de charge et la quantité de charge.

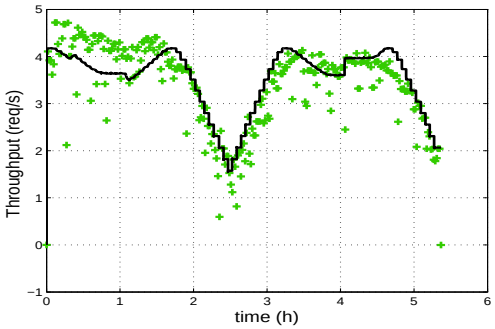
La figure 4 décrit le premier scénario dans lequel le mix de charge varie de M1 à M2 puis revient à M1 (c.f. figure 4(a)), tandis que la quantité de charge est de 80 clients. Le mix de charge M1 est constitué de requêtes en lecture-écriture, M2 de requêtes en lecture seule. Les figures 4(b), 4(c) et 4(d) présentent les évolutions respectives dans le temps du MPL, de la latence moyenne et du taux de rejet, pour le système contrôlé et le système non contrôlé. On remarquera que la variation soudaine du MPL aux vingtième et quarantième minutes correspond aux changements de mix de charge, ce qui influe aussi sur la latence et le taux de rejet.



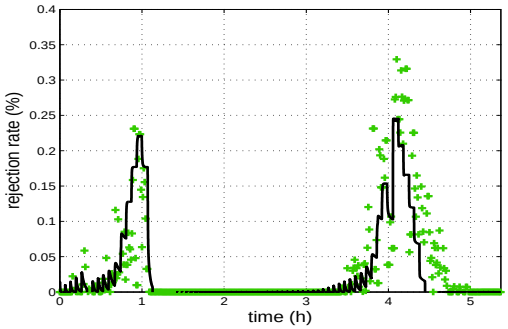
(a) Variation du MPL et de la quantité de charge



(b) Clients concurrents admis

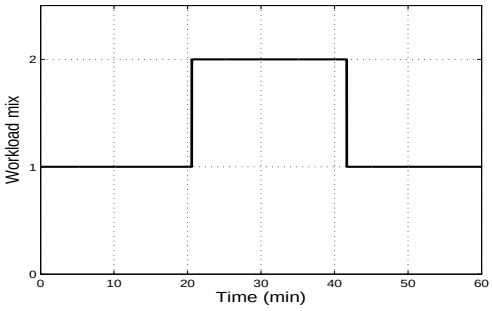


(c) Débit

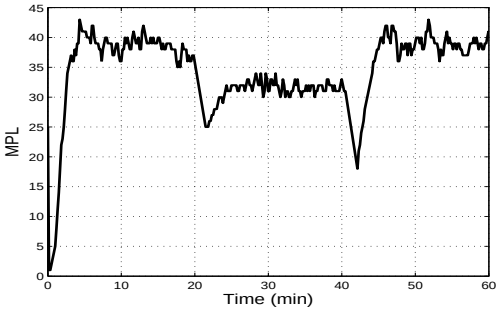


(d) Taux de rejet

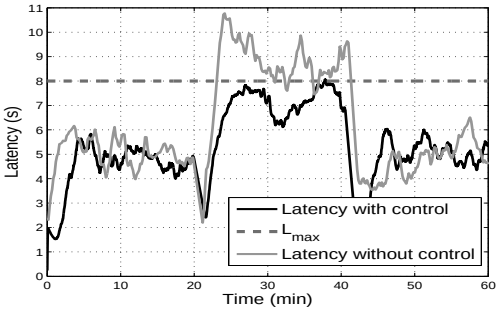
FIG. 3 – Comportement du système pour un MPL et une charge variables – système réel (+) et système modélisé (-)



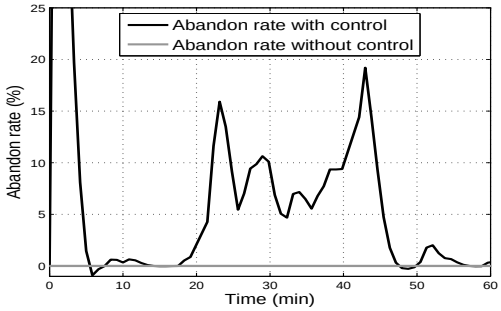
(a) Mix de charge



(b) MPL du système contrôlé



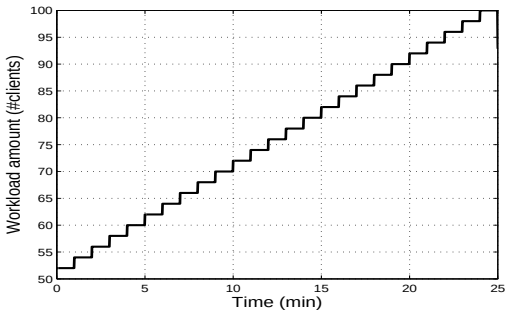
(c) Latence



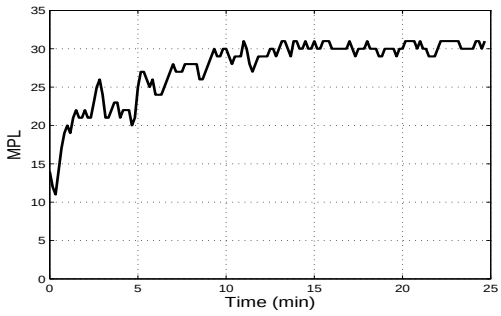
(d) Taux de rejet

FIG. 4 – Comportement du système pour une variation du mix de charge – système contrôlé par AM-$\mathscr{C}$ et système non contrôlé

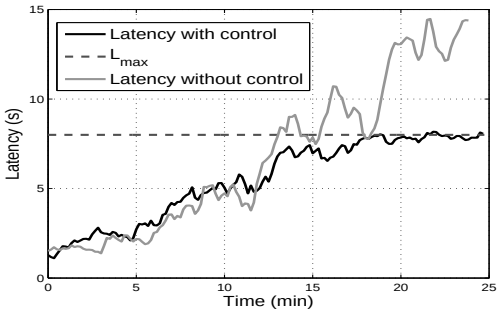
Les résultats montrent que la commande AM-$\mathscr{C}$ est capable de garantir une contrainte sur la latence tout en conservant une disponibilité de service maximale, avec un taux de rejet minimisé à 0% avec M1 et 9% en moyenne avec M2. Pour le système non contrôlé, où la concurrence sur le serveur peut être illimitée, la QoS n'est pas garantie et la performance se dégrade avec un dépassement de 30% sur la latence.
La figure 5 présente le comportement en boucle fermée du système pour une variation de la quantité de charge dans le temps (c.f. figure 5(a)) quand le mix de charge est constant à M1. Les figures 5(b), 5(c) et 5(d) montrent les évolutions respectives dans le temps du MPL, de la latence moyenne et du taux de rejet, pour le système contrôlé et le système non contrôlé. On remarquera qu'à cause du "think time" du client TPC-C, le nombre de clients actifs à un instant donné peut être inférieur à la charge générée par l'émulateur. Les résultats montrent que pour le système contrôlé, le MPL est ajusté à la valeur optimale qui garantit la contrainte de performance tandis que pour le système non contrôlé, la latence atteint 14.4s, ce qui représente un dépassement de 80% par rapport au système contrôlé.



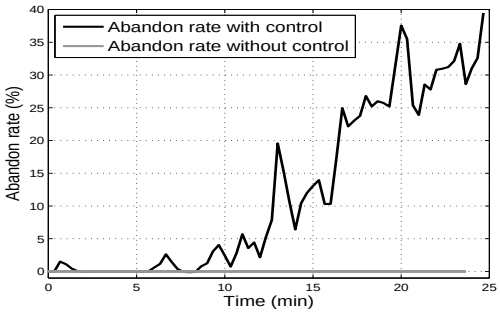
(a) Quantité de charge



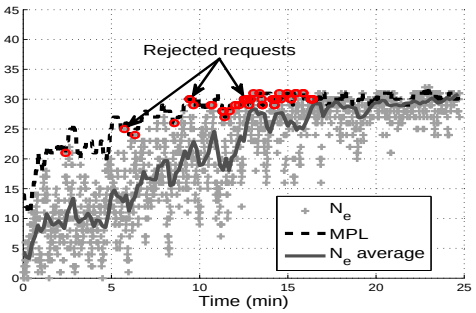
(b) MPL du système contrôlé



(c) Latence



(d) Taux de rejet



(e) Client concurrents admis (N_e) vs. MPL

FIG. 5 – Comportement du système pour une variation de quantité de charge – système contrôlé par AM-$\mathscr{C}$ et système non contrôlé

Pour le système contrôlé, le taux de rejet reste inférieur à 5% jusqu'à une charge de 75 clients, puis augmente avec la quantité de charge afin de permettre au système de maintenir une latence inférieure à la contrainte

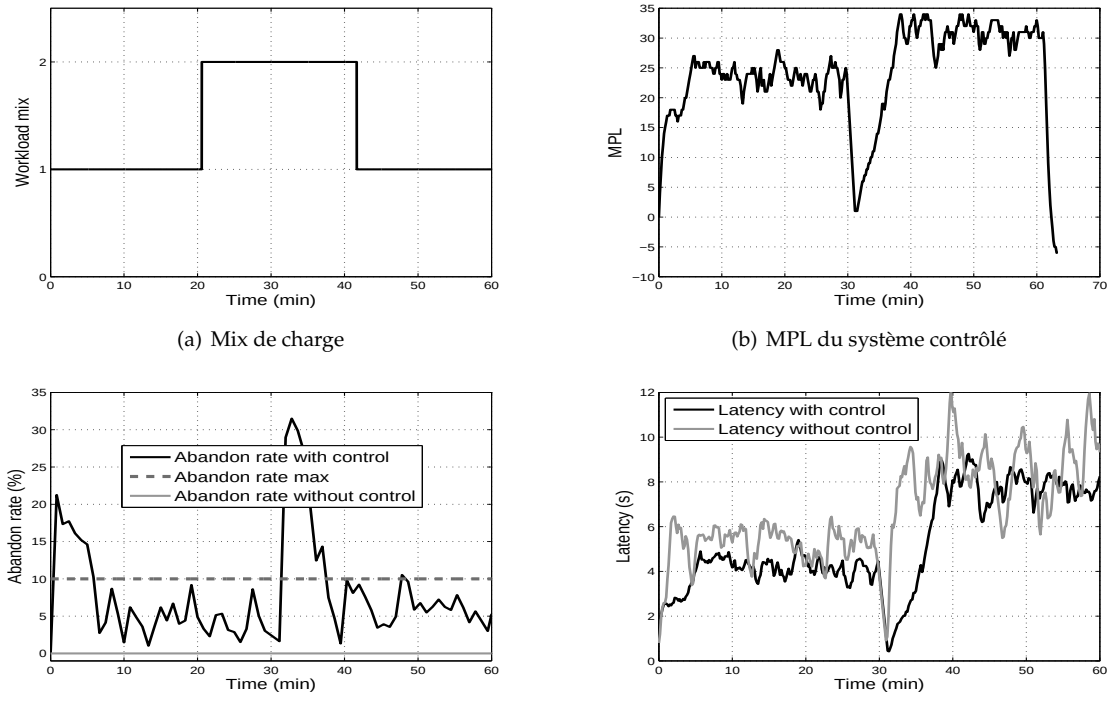
maximale. Il est justifié d'avoir un fort taux de rejet à la fin de l'expérience (entre 18 et 25 minutes), puisque la latence a atteint sa valeur maximale et la quantité de charge continue à augmenter. Cependant, le fait que le taux de rejet soit supérieur à 0% dans la première moitié de l'expérience alors que la latence est inférieure à sa valeur maximale autorisée doit être soulevé. On peut l'expliquer figure 5(e) par la nature aléatoire de la charge qui, à un instant donné, peut présenter une quantité de charge supérieure au MPL tandis que le nombre moyen de clients dans le système N_e est inférieur au MPL. Ceci génère du rejet à ces différents instants, représentés par des cercles dans la figure 5(e). Afin de résoudre ce problème, une commande améliorée consisterait à prendre en compte les situations de sous-charge en autorisant les nouvelles connexions tant que la contrainte de latence est garantie. Le manque d'espace nous empêche de présenter les résultats de cette amélioration.

5.3.2. PM-$\mathscr{C}$ commande en performance maximale

Dans cette section nous évaluons la commande en performance maximale PM-$\mathscr{C}$ proposée en section 4.2. Nous considérons ici une contrainte de disponibilité de service qui limite le taux de rejet à 10%. Le rôle de PM-$\mathscr{C}$ est alors de garantir cette contrainte de disponibilité tout en maximisant la performance du service grâce à un ajustement en ligne du MPL.

La figure 6 présente l'évolution du système et de la commande pour une variation du mix de charge en entrée [1]. Dans la figure 6(a) on voit que le mix de charge varie de M1 à M2 puis revient à M1. La quantité de charge N est de 80 clients. Les figures 4(b), 4(c) et 4(d) présentent les évolutions respectives dans le temps du MPL, de la latence moyenne et du taux de rejet, pour le système contrôlé et le système non contrôlé. Ici encore on remarquera que la variation soudaine du MPL aux vingtième et quarantième minutes correspond aux changements de mix de charge.

Les résultats montrent que la commande PM-$\mathscr{C}$ est capable de garantir une contrainte sur le taux de rejet tout en conservant une performance service maximale, avec une latence moyenne minimisée à 4s pour M1 et 8s pour M2. La latence du système contrôlé est de 20% inférieure à celle du système non contrôlé.



(a) Mix de charge

(b) MPL du système contrôlé

(c) Taux de rejet

(d) Latence

FIG. 6 – Comportement du système pour une variation du mix de charge – système contrôlé par PM-$\mathscr{C}$ et système non contrôlé

[1] Par manque d'espace les résultats de l'évaluation avec une variation de la quantité de charge ne sont pas présentés

6. Etat de l'art

La performance et la disponibilité des systèmes de serveurs sont intimement liées à leur configuration [19, 20]. De nombreux travaux ont été menés dans le domaine de la gestion de la QoS des systèmes. Des techniques de contrôle d'admission, de différentiation de service, de dégradation de service et d'ordonnancement des requêtes ont été étudiées. Par manque de place, nous ne présenterons qu'une vue d'ensemble des travaux concernant le contrôle d'admission pour la gestion des systèmes de serveurs.

Alors que l'amélioration des performances et de la disponibilité des systèmes de serveur est habituellement gérée par des administrateurs humain qui utilisent des réglages ad-hoc [6, 22], de nouvelles approches sont apparues afin de simplifier la gestion de ces systèmes. Menascé et. al. proposent une heuristique pour la gestion de la QoS à l'aide de la détermination de leur niveau de multiprogrammation (MPL) en utilisant une technique d'optimisation du type hill-climbing [21]. Bien qu'elle donne de bons résultats dans de nombreuses applications, cette technique ne garantit pas l'optimalité de la solution. Dans [9], une technique similaire est appliquée ; cependant le MPL est déterminé hors-ligne et par conséquent, ne permet pas de s'adapter aux variations de charge. D'autres solutions d'identification du MPL sont proposées pour des technologies spécifiques, telles que les serveurs transactionnels [27].

Il existe des approches qui visent à modéliser le système pour en caractériser sa capacité. Dans [10], une étude fondée sur des simulations est menée et un modèle analytique pour ajuster le MPL en fonction des changements de charge est proposé. Ce modèle ne s'applique malheureusement qu'à des critères de performance en forme de parabole, ce qui n'est pas le cas de la latence ou du taux de rejet, qui sont des critères de QoS directement perçus par les clients. D'autres travaux visant à appliquer la théorie de la commande sont apparus dans la dernière décennie. Une première approche consiste à utiliser des technique connues de commande linéaire sur des serveurs modélisés par des boites noires SISO (une entrée, une sortie) ou MIMO (plusieurs entrées, plusieurs sorties) [26, 7, 11]. Bien que localement ces techniques peuvent s'avérer intéressantes, le comportement non linéaire intrinsèque d'un serveur ne leur permet pas d'être efficace sur toute la plage de charge à laquelle il peut être soumis dans un cas réel. Des approches se basent sur des modèles non linéaires issus de la théorie des files d'attente [29, 31] avec des propositions théoriques dans [16, 17, 25]. Les modèles qui en découlent sont capables de prédire de manière pertinente la performance du système, mais au prix d'une calibration délicate de leur paramètres.

Le système AM/PM-$\mathscr{C}$ proposé diffère des travaux précédents à de nombreux égards. Il met en oeuvre des techniques de commande non linéaire basées sur un modèle issu d'une approche de type fluide, contenant peu de paramètres externes. Les approximations de type fluide sont utilisées avec succès pour modéliser et contrôler des systèmes dans d'autres domaines d'applications tels que la circulation routière et les modèles de population. Dans le travail proposé, nous appliquons cette approche pour modéliser et contrôler un serveur. Nous montrons comment cela permet de garantir à la fois une performance et une disponibilité du service.

7. Conclusion

Cet article présente la construction, l'implémentation et l'évaluation d'un modèle continu non linéaire de serveur basé sur des approximations de type fluide, duquel on déduit un contrôle d'admission optimal pour un compromis performance-disponibilité donné. Deux lois de commande pour le contrôle d'admission sont proposées pour deux objectifs de QoS. AM-$\mathscr{C}$ garantit une contrainte de performance en maximisant la disponibilité du service. PM-$\mathscr{C}$ garantit une contrainte de disponibilité du service en maximisant la performance. Nos expériences montrent que les techniques proposées peuvent améliorer la performance d'un serveur de 30% tout en garantissant une contrainte de disponibilité. Alors que cet article est orienté sur des métriques de QoS telles que la latence et le taux de rejet, nous pensons que le modèle proposé et l'approche utilisée pour la commande peuvent s'appliquer à d'autres métriques, telles que le débit du serveur. Bien que les techniques proposées aient été appliquées à un serveur de base de données, nous pensons qu'elles pourraient facilement être employées sur n'importe quel système de serveur où le contrôle d'admission est possible (comme les seveurs web ou les serveurs d'application). En outre, nous nous intéressons à la possibilité de les appliquer à des systèmes distribués.

Bibliographie

1. T. Abdelzaher, Ying Lu, Ronghua Zhang, and D. Henriksson. Practical application of control theory to Web services. *American Control Conference*, June 2004.

2. Amazon.com Inc, 2007. http ://www.amazon.com/.
3. Apple Inc. QuickTime Streaming Server, 2007. http ://www.apple.com/quicktime/streamingserver/.
4. Martin Arlitt and Tai Jin. Workload Characterization of the 1998 World Cup Web Site. Technical Report HPL-1999-35(R.1), HP Laboratories Palo Alto, September 1999.
5. Martin Arlitt and Carey L. Williamson. Web server workload characterization : the search for invariants. *SIGMETRICS Perform. Eval. Rev.*, 24(1) :126–137, 1996.
6. Martin Brown. Optimizing Apache Server Performance, February 2008. http ://www.serverwatch.com/tutorials/article.php/3436911.
7. Yixin Diao, N. Gandhi, J.L. Hellerstein, S. Parekh, and D.M. Tilbury. Using MIMO feedback control to enforce policies for interrelated metrics with application to the Apache Web server. *Network Operations and Management Symposium*, 2002.
8. John A. Dilley. Web Server Workload Characterization . Technical Report HPL-96-160, HP Laboratories, December 1996.
9. S. Elnikety, E. Nahum, J. Tracey, and W. Zwaenepoel. A Method for Transparent Admission Control and Request Scheduling in E-Commerce Web Sites. In *13th international conference on World Wide Web*, New York, NY, May 2004.
10. Hans-Ulrich Heiss and Roger Wagner. Adaptive Load Control in Transaction Processing Systems. In *17th International Conference on Very Large Data Bases*, San Francisco, CA, 1991.
11. J. Hellerstein, Y. Diao, S. Parekh, and D.M. Tilbury. *Feedback Control of Computing Systems*. John Wiley & Sons, 2004.
12. J. Hyman, A. A. Lazar, and G. Pacifici. Joint Scheduling and Admission Control for ATS-based Switching Nodes. In *ACM SIGCOMM)*, Baltimore, MA, August 1992.
13. Iron Mountain. The Business Case for Disaster Recovery Planning : Calculating the Cost of Downtime, 2001. http ://www.ironmountain.com/dataprotection/resources/ CostOfDowntimeIrnMtn.pdf.
14. Rudolph Emil Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D) :35–45, 1960.
15. Hassan Khalil. *Nonlinear Systems*. Prentice Hall, 2002.
16. M. Kihl, A. Robertsson, and B. Wittenmark. Analysis of admission control mechanisms using non-linear control theory. *8th IEEE International Symposium on Computers and Communication*, pages 1306–1311 vol.2, July 2003.
17. M. Kihl, A. Robertsson, and B. Wittenmark. Performance modelling and control of server systems using non-linear control theory. In *18th International Teletraffic Congress*, Berlin, Germany, September 2003.
18. J. D. C. Little. A proof for the queueing formula L $=$ λ W. *Operation Research*, 9 :383–387, 1961.
19. C. Loosley, F. Douglas, and A. Mimo. *High-Performance Client/Server*. John Wiley & Sons, 1997.
20. Evan Marcus and Hal Stern. *Blueprints for High Availability*. Wiley, September 2003.
21. D. A. Menascé, D. Barbara, and R. Dodge. Preserving QoS of E-Commerce Sites Through Self-Tuning : A Performance Model Approach. In *ACM Conference on Electronic Commerce*, Tampa, FL, October 2001.
22. Microsoft. Optimizing Database Performance. http ://msdn.microsoft.com/en-us/library/aa273605(SQL.80).aspx.
23. North American Systems International Inc. The True Cost of Downtime, 2008. http ://www.nasi.com/downtime_cost.php.
24. PostgreSQL, 2008. http ://www.postgresql.org/.
25. A. Robertsson, B. Wittenmark, M. Kihl, and M. Andersson. Admission control for web server systems - design and experimental evaluation. *43rd IEEE Conference on Decision and Control*, December 2004.
26. S. Parekh and N. Gandhi and J. Hellerstein and D. Tilbury and T. Jayram and J. Bigus. Using Control Theory to Achieve Service Level Objectives In Performance Management. *Real-Time Syst.*, 23(1-2) :127–141, 2002.
27. Bianca Schroeder, Mor Harchol-Balter, Arun Iyengar, Erich Nahum, and Adam Wierman. How to determine a good multi-programming level for external scheduling. In *22nd International Conference on Data Engineering*, Atlanta, GA, April 2006.
28. Sendmail.org, 2007. http ://www.sendmail.org/.
29. D. Tipper and M.K. Sundareshan. Numerical methods for modeling computer networks under nonstationary conditions. *IEEE Journal on Selected Areas in Communications*, 8(9) :1682–1695, December 1990.
30. TPC-C. Tpc transaction processing performance council, 2008. http ://www.tpc.org/tpcc/.
31. Wei-Ping Wang, D. Tipper, and S. Banerjee. A simple approximation for modeling nonstationary queues. *IEEE INFOCOM*, March 1996.